Summary of “Transferring Unconditional to Conditional GANs with Hyper-Modulation”

Abstract

-Problem: huge time & data needed to train GANs. Importance of transfer learning to deal with that problem.

-Many high-quality pretrained GANs exist, but are unconditional

-conditional GANs are preferred because they provide more control over generated output

-Goal of this paper is to transfer from "high-quality pretrained unconditional GANs to conditional GANs" (turn GAN into cGAN)

-paper proposes "hyper-modulated generative networks that allow for shared and complementary supervision"

-with that method, they fight overfitting with a "self-initialization procedure that does not require any real data to initialize the hypernetwork parameters"

-this paper also applies "contrastive learning in the discriminator"

Introduction

-GANs take a lot of time and resources to train

-Another source "investigated unconditional transferring by fine-tuning a pre-trained GAN to a target domain"

-More ways to improve transfer learning to small domains: "reducing the number of learnable parameters or by identifying the subspace of a pre-trained GAN that best models the target data"

-Most experiments on transfer learning are from (1) unconditional GAN to unconditional GAN, from (2) conditional to unconditional (done by "transferring a pre-trained cGAN to a single-class target domain"), or from (3) conditional to conditional (done "through linear combination of conditionings")

-This paper will look at (4) knowledge transfer from unconditional to conditional

-Aside from being useful for being able to control what's generated, another benefit of transferring to cGANs ("compared to transferring to multiple unconditional GANs") is that it also enables "the sharing of weights between the multiple classes, thereby exploiting the similarities between the various classes"

Introduction (cont.)

This paper proposes to "leverage weight modulation from the context of continual learning to transform an unconditional source GAN to a cGAN"

-"frozen pre-trained weights are conditionally modulated to yield target-specific outputs" (drawback: "class-specific modulation parameters are learned independent of each other")

-use hypernetworks to "exploit the existing similarities among the multiple classes of the target domain"

-using hypernetworks in transfer learning = novel, according to this paper

-problem: hypernetwork "introduces new parameters that need to be trained from scratch" the paper's solution to initialize these parameters is a "self-alignment method that learns well-initialized hypernetworks without getting access to any real data"

-finally, they add "contrastive learning in the discriminator for quality improvement" (but they say this works only with very limited batch sizes like 10 samples)

Related Work

-GANs

-Transfer learning

-Hypernetworks

    -Hypernetworks = "implicit generators that aim to generate parameters for other models"

-Contrastive learning

    -"bridging the gap between supervised and unsupervised learning"

Methodology: Overview

Given a source domain and a multi-class target domain, along with a pre-trained model on the source domain, this paper aims to "use transfer learning to efficiently learn a hypernetwork that can generate weights for all classes of the target domain"

1) introduce "class specific parameters that result in a certain modulation of the forward pass through the generator"

2) propose "hypernetworks to directly estimate the modulation parameters – and importantly share the knowledge required to generate them among the classes."

3) new self-distillation method to learn weights for hypernetworks without needing data

4) contrastive learning to aid efficiency and quality of model

1. Domain transfer

- Start with a source domain (i.e. a pretrained model) called h(x), and h(x) = Wx + b where W are pre-trained weights and x is the input
- Then, form a new layer with equation:

$$\hat{\boldsymbol{W}}_i = \boldsymbol{\gamma}_i \odot \frac{\boldsymbol{W} - \boldsymbol{\mu}}{\boldsymbol{\sigma}} + \boldsymbol{\beta}_i, \quad (1)$$

$$\hat{\boldsymbol{b}}_i = \boldsymbol{b} + \boldsymbol{b}_i, \quad (2)$$

where $\boldsymbol{\gamma}_i, \boldsymbol{\beta}_i \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ are learned parameters, $i = 1, ..., N_c$ indicates the class, N_c is the number of classes, and $\boldsymbol{\mu}, \boldsymbol{\sigma}$ are the mean and standard deviation of $\boldsymbol{W}_i$.

- γ, β, b are the modulation parameters that change in the layer, while pretrained network weights W & b (shared between classes) stay the same.

(modulation parameter refers to the parameter influencing the result)

1. Domain transfer (cont.)

- Paper cites another source that "this modulation allows to model large domain shifts" (so this method of changing modulation parameters is valid)
- So, conditioning those parameters can lead to producing “conditional networks from an unconditional base”

Example: image shows what each modulation parameter does in the “knowledge transference”

what the modulation did: “first removes the source style encoded in μ,σ and then apply the learned one from γ, β to model the statistics of a generative process of the target distribution”

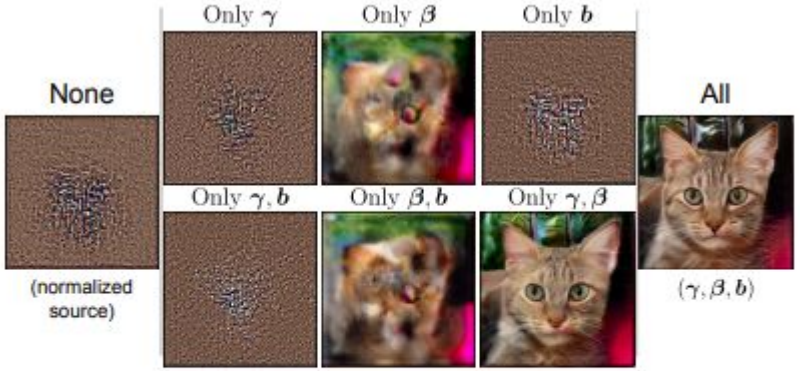

Figure 2. Effect of the modulation parameters on the domain transfer. Parameters γ generate high-frequency details, *i.e.*, texture and structure. β takes care of low-frequency details, *i.e.*, color. b is for localized details unattainable otherwise. A detailed figure is shown in the Appendix.

2. Hyper-modulation

-Domain Transfer’s shortcoming = the modulation is "optimized for each class in the target domain separately". Meaning they just do each class separately. And it's inefficient because there are similarities between classes that can be exploited. They use hypernetworks to fix this.

-Definition: neural network = family of functions that learns parameters Θ (weights, biases)

-Definition: hypernetwork = neural network that learns the parameters Θ_h of "a metamodel, which will then generate the target parameters Θ_{trg} of the target model” (so hypernetwork generates the target weights)

-in this paper: use hypernetwork to predict the modulation parameters conditionally (those 3 modulation parameters, y B and b, that changed in the equation), which will eventually “[enable] us to produce a generative model for each target”

2. Hyper-modulation (cont.)

- Input to the hypernetwork: "a vector coming from a class embedding network $C(i; \Psi) = v \in V$ where $(i = 1, ..., N_c)$ is the class label, V is the class embedding space, and Ψ are network parameters."
- "By varying the number of parameters Ψ, we are able to vary the class knowledge capacity of the system. The hypernetwork g takes the embedding vector v and maps it to the modulation parameters according to:"

$$\gamma_{\mathbf{v}}, \beta_{\mathbf{v}} = g(\mathbf{v}; \Phi_a), \qquad \boldsymbol{b}_{\mathbf{v}} = g_b(\mathbf{v}; \Phi_b) \qquad (3)$$

where g are affine projections of a point in the space $\mathcal{V}$, with network parameters Φ_a and Φ_b. We use $\boldsymbol{\Phi}$ to denote the combination of all the parameters used by the hypernetwork, consisting of Φ_a and Φ_b for all the layers in the network. Each modulated layer has a g projector, but layer-wise, these are shared among target classes.

2. Hyper-modulation (cont.)

“modulation that produces target-specific activations” defined as $h_v(x)$:

$h_{\mathbf{v}}(\boldsymbol{x}) = \hat{\boldsymbol{W}}_{\mathbf{v}}\boldsymbol{x} + \hat{\boldsymbol{b}}_{\mathbf{v}}$ is of the form

$$\hat{\boldsymbol{W}}_{\mathbf{v}} = \boldsymbol{\gamma}_{\mathbf{v}} \odot \frac{\boldsymbol{W} - \boldsymbol{\mu}}{\boldsymbol{\sigma}} + \boldsymbol{\beta}_{\mathbf{v}}, \tag{4}$$

$$\hat{\boldsymbol{b}}_{\mathbf{v}} = \boldsymbol{b} + \boldsymbol{b}_{\mathbf{v}}, \tag{5}$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ are the frozen source weights. Ultimately, a hypermodulator f will be given a class embedding $\mathbf{v}$ and a normalized source weight $\tilde{\mathbf{w}}$ to produce the desired target weights as $f_{\tilde{\mathbf{w}}}(\mathbf{v}) = \boldsymbol{\gamma}_{\mathbf{v}} \odot \tilde{\mathbf{w}} + \boldsymbol{\beta}_{\mathbf{v}} = \hat{\mathbf{w}}_{\mathbf{v}}$, following Eqs. (3) and (4) and pictured in Fig. 4.

-above is just the modulation equation (very similar to equation shown in Domain Transfer section except ‘i’ is now ‘v’, the class vector); a hypermodulator f will be used to take vector input of classes + normalized source weight and then calculate “desired target weights”

3. Self-alignment

-An effect of the introduction of the new modules = some previously learned stuff/weights in the model disappear because “the parameters Ψ, Φ have not been learned yet” (those params were in the hypernetwork section, shown again below)

$$\gamma_{\mathbf{v}}, \beta_{\mathbf{v}} = g(\mathbf{v}; \Phi_a), \qquad \boldsymbol{b}_{\mathbf{v}} = g_b(\mathbf{v}; \Phi_b) \qquad (3)$$

where g are affine projections of a point in the space $\mathcal{V}$, with network parameters Φ_a and Φ_b. We use $\boldsymbol{\Phi}$ to denote the combination of all the parameters used by the hypernetwork, consisting of Φ_a and Φ_b for all the layers in the network. Each modulated layer has a g projector, but layer-wise, these are shared among target classes.

-That’s OK because it’s just the process of new classes being learned. But affects general training times because network needs to relearn some knowledge. So to reduce training time, they try to return those lost weights with a self-alignment initialization

3. Self-alignment (cont.)

-Proposed solution: self-align the not-learned-yet parameters Ψ, Φ

-this not only “[recovers] the original weight statistics”, but also initializes a “sensible latent space for the embedding vectors v that could be further augmented by new classes”

-actual operation: "does not require any real data, since we can align the two networks by simply sampling random vectors z"

"the self-alignment initializes the hypernetwork parameters Ψ, Φ. When we now finetune the network on the multi-class target domain, we do not have to learn these parameters from scratch."

-result of self-alignment = faster training time & better generated result quality

4. Contrastive learning

-applied contrastive learning to discriminator during adversarial training

-making discriminator better = it can distinguish fake images better = it can make the generator try to make even better images, and as a result, we get better image quality from the generator

-Barlow Twins

Experiments

- method was applied to pre-trained StyleGAN
- They changed the top layer of the discriminator by replacing the last fully connected layer with a “convolutional layer with 3 x 3 filter size” that has an “output channel dimensionality of [number of classes in target domain]”
- kept hyperparameters from the original model the same (such as Adam and R1 regularization), and model was trained at 256x256 resolution

Experiments (cont.)

- Experimented on multiple different datasets:
  - 3 classes, each 5000 images
  - 2 classes, with 10,000 and 18,000 images for the 2 classes
  - an unconditional dataset (ignored labels)
  - Places 365 dataset with only 10 selected categories as target

- all images resized to 256x256 for experiments

Experiments (cont.)

-mention little existing previous work to properly compare against (since not much work exploring transfer learning from gans to cgans)

-general results from comparing some older modulations to this paper's proposed hypernetwork:

- "better synthesis quality", higher diversity; improved recall & coverage

-paper attributes these better results to “the knowledge sharing and complementary supervision in the joint training, since each input is affecting and shaping the whole hypernetwork as opposed to learning separate embedding points for modulation”

Experiments (cont.)

- Self-initialization helped training time and quality (they compared training with uninitialized hypernetwork vs a self-aligned hypernetwork)

- Contrastive learning - paper found that self-supervision = beneficial for transfer learning. However, contrastive learning in the generator did not improve quality

Experiments (cont.)

-Performed experiments on close vs far domain transfer

-close domain: small domain gap; used domains of animal face and human face (Used a pretrained animal-face-optimized styleGAN when target domain is human face, and vice versa)

-far domain: used pretrained human-face-optimized gan and two target domains of 1) flowers, and 2) places (Places365 dataset)

-for both, proposed method was an improvement compared to the baseline "GAN Memory" (unconditional to unconditional) when comparing FID scores

Conclusions

-proposed hyper-modulation "to produce weight modulation parameters on-the-fly for a source model"

-added self-initialization method (which didn't need any data to learn well-initialized weights) to speed up training

-added self-supervision to discriminator to further improve it

-overall, proposed method outperforms current results on transfer learning

-paper notes limitation of memory, that in the future they could try to not keep the whole pre-trained network in memory

Reference

Laria, H., Wang, Y., van de Weijer, J., & Raducanu, B. (2022). Transferring unconditional to conditional GANs with hyper-modulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 3840-3849).